AN INTRODUCTION TO GRAPHICS AND GRAPHIC FILES

Your eye is the single most sophisticated image digitizing mechanism in the known universe – at least, it is unless you bought one of those cheap monitors from Sears and habitually sit too close to it. One of the remarkable things about your eye is a level of user-friendliness that would make the user-friendliness of even the most over-marketed software look like naked aggression by comparison. It's so easy to operate, it doesn't even come with instructions.

The visual world is analog, which is to say that real-world images are not comprised of discrete bits of information unless you want to get down to the level of quantum physics. Every image embodies an infinite number of details. This makes for a really interesting visual world, but it's a problem for sensory mechanisms that want to perceive it. Infinity is uncomfortably large. All sensory mechanisms want to deal with a finite number of discrete objects – they are required to split the images they perceive into little bits of light.

Your eye does this by using a blanket of photosensitive cells – those of us who didn't cut high school biology will recall these as being called "rods" and "cones." They split up the analog world into a lot of little bits of information, which your brain can subsequently make sense of. While your brain is sophisticated enough to filter all this into a perception of an analog world, what it really sees through your eyes is a matrix of very small colored dots.

A computer that wanted to represent visual reality would be confronted with the same problem as that of your eye, and its designers would solve it in much the same way. Rather than saying "let there be light" or "give it twenty-million years and let's see what evolves," computer designers simply copied the way eyes work into a digital equivalent. They created bitmaps.

A bitmap is a matrix of colored dots. Computer-based bitmaps are a lot coarser than the bitmaps produced by your eyes, of course, because they're being handled by a computer billions of times less powerful than the one your eyes are connected to. In addition, the one your eyes are connected to isn't likely to be recalled because of a floating point divide bug – extensive beta testing has its advantages.

This should help illustrate the nature of a bitmap. The left side of the picture appears to be an analog image, that is, what your eye might see were it to be pointed at the image in question.

The right side illustrates how the image is really constructed. The colored dots that make it up are too small to see when you view the image in its entirely, but they're present none the less.

Most technologies that display images can be seen to be bitmaps if you look carefully. Your monitor is one, as is a television set. Photography is also a bitmapped technology – its matrix of dots is actually a matrix of silver salt crystals, so the dots are exceedingly fine. Computer-based imaging can't approach the resolution of either chemical photography or of the biological camera in your head, but because we have fairly tolerant brains, it rarely has to try.

The colored dots that make up a bitmap are properly called "pixels" – unless you work for IBM, in which case they're called "pels." If you're uncertain whether you should be referring to them as pixels or pels, buy a small potted plant and leave it on your desk for a few days. If the plant is confiscated by your boss, refer to them as pels.

In addition to providing your eyes with infinitely small details, the analog world presents it with images having the potential for an infinite number of possible colors. Again keeping in mind that ideas like "infinite" really upset computer designers, digital bitmaps can represent a large but finite number of colors. For reasons we'll get to in a moment, the bitmaps that Graphic Workshop and most other imaging software deal with can represent a maximum of 16,777,216 possible colors. It's this rather large number, and a bit of calculator action, that will begin to explain why bitmapped graphic files are so uncomfortably large.

Color is represented on a computer by using varying amounts of red, green and blue light. These are the primary colors of what's called "additive" color – by adding percentages of red, green and blue to the initial blackness of your monitor, any color can be created.

In the simplest sort of bitmapped image, each pixel is represented by three numbers to store the amounts of red, green and blue light that define the color of the pixel in question. The smallest useful object for storing numbers on a computer is a byte – in this sort of bitmap, each pixel requires one byte for each color index for a total of three bytes per pixel. As a byte represents eight bits, each pixel requires 24 bits to store all its color information. This defines the maximum number of discrete colors this sort of bitmap can represent as 2 raised to the power of 24, or 16,777,216.

Such graphics are referred to as "true-colour" images, or just as "24-bit" graphics.

You needn't remember these numbers – there won't be a quiz at the end of this document.

Here's where you can use the aforementioned calculator. The lowest resolution for a monitor displaying a Windows desktop is 640 by 480 pixels. In a bitmap of this resolution, then, there would be three bytes per pixel, for a total of 640 x 480 x 3 bytes, or about 900 kilobytes. A few graphics of this size would require more hard drive space to store them than all the executable parts of the Graphic Workshop software.

This is the fundamental issue out of which arises almost all the complexities which are addressed by graphic file formats. Bitmapped graphics are huge entities, and they become huger still as they get better looking. There are ways to reduce the level of hugeness represented by bitmaps, but they involve all sorts of stealth and treachery. Graphic file formats are the repositories of said stealth and treachery.

Because hard drive space and the bandwidth over which images are transmitted across the internet are finite and relatively expensive commodities, it's desirable to make the files which store bitmaps as small as possible. In a simple sense, it's very often realistic to compress a raw bitmapped graphic into a smaller file for storage or transmission, and subsequently uncompress it to view or manipulate it. A number of remarkably sneaky ways have been devised to compress images.

You probably use data compression all the time – it's the basis of archiving packages such as PKZIP and of the hardware data compression used by modems. Data compression works by finding areas in your graphics which are all the same colour and replacing them with notes which say, in effect "this area is all the same colour." Allowing that these notes require less space than the areas of your graphics they replace, your graphics will be compressed into files smaller than the graphics themselves would require.

The catch to this sort of compression is that really complex graphics frequently exhibit very few areas which are all the same colour. Photorealistic images – as a rule, the most interesting graphics – typically have almost no compressible areas, and often result in really large files.

The better they look, the bigger they're likely to get.

Graphic file formats which contain bitmaps can be regarded as being of three types. The first type stores graphics uncompressed – the final file size will typically be a bit larger than the size of the graphic stored in it, allowing for a bit of extra space for internal housekeeping for the file in question. Windows BMP files are an example of this sort of format.

The second type uses some variation on the type of compression we've just discussed. These are called "non-lossy" formats. Whatever gets stored in them will be identical to what emerges from a non-lossy file when it's unpacked. Most graphic formats use non-lossy compression – the GIF and PCX formats are among them. The catch to non-lossy compression is that it's not usually all that effective, and if you find that you need more compression than the format in question can manage, there's not much you'll be able to do about it.

The third type of bitmapped graphic file formats uses, perhaps not surprisingly, "lossy" compression. Photorealistic images don't compress well because they have lots of details – the

details are what prevent areas from being all the same colour, and as such from responding well to compression. In some cases the details represent very subtle colour variations – perhaps too subtle to be discernable by your eye, or at least, too subtle to make much of a difference to your perception of the graphic in question.

Lossy compression seeks to improve upon the compression of the graphics it deals with by throwing away some of the details in your source images to create more areas which are all the same colour. Having done so, lossy compression can typically improve upon the file compression offered by the non-lossy formats.

One of the obvious catches in lossy compression is that the amount of detail which can be removed from a graphic before it starts looking ugly and causes nearby bacteria mutate is a highly subjective matter. Clearly, it's nothing you'd want to leave to software to decide. As such, lossy compression allows you to specify the amount of detail to be discarded when a graphic is compressed. This level of image degradation is given the most positive spin possible – it's called the "quality factor."

The most commonly encountered lossy image file format is JPEG – Graphic Workshop supports several others, including ART, FIF, PIC, KQP and FlashPix. The section of this document which deals with JPEG files will get into the dark and turgid secrets of lossy compression a little later.

Graphic File Formats

Every bitmapped graphics that resides on your hard drive or oozes over the internet must be stored in a file format. There are dozens of extant file formats, every one of them structured differently. Graphic Workshop reads over fifty of these – there are several times this number of other graphic file formats about, in most cases created for use with specific applications. These latter formats are usually regarded as being "proprietary" – their creators have either deliberately chosen not to release the details of how they're structured, or they just never got around to doing so. To this end, applications such as Graphic Workshop can't handle most proprietary formats.

Some formats are fairly common – if you surf the web, you'll no doubt have encountered lots of GIF and JPEG files, the common image file formats of cyberspace. Windows stores its wallpaper in the BMP format. Users of America On Line often encounter graphics in the ART format. If you get your photographs developed and provided on floppy disk, you'll probably be confronted with graphic files in the PIC or KQP formats.

Graphic files are usually stored with file names that indicate the format they're stored in. As such, a picture stored as a GIF file might be called PICTURE.GIF. A picture stored as a BMP file could be named, perhaps predictably, PICTURE.BMP. By convention, file extensions – the letters after the dot in a file name – are limited to three letters. As such, JPEG files are usually named with the extension .JPG – you'll occasionally encounter JPEG files with the extension .JPEG or .JPE as well.

Now and again you'll come across graphic files with incorrect extensions, or no extension at all. The mystery graphic format identifier function of Graphic Workshop can be useful in figuring out what these files really are.

For the sake of this discussion, the same images can be stored in all these formats, and many others. Graphic workshop can convert between graphic file formats – for example, if you wanted to import a picture stored as an AOL ART file into a Microsoft Word document, you might use Graphic Workshop to convert it to BMP, a format that Word knows how to deal with.

There's a much more extensive discussion of the formats Graphic Workshop can support in the Formats document.

If you encounter problems in using graphic files, they will most likely arise from the inability of the software you want to use them with to read the particular graphic file format your graphics are stored in. Most applications which can open or import bitmapped graphics can only do so for a relatively small number of formats. This is where the format conversion facilities of Graphic Workshop can bail you out of considerable trouble.

In should be mentioned that there is a second broad classification of graphic file formats. In addition to bitmapped graphics – also called "raster graphics" in some circles – there are vector or "line art" graphics about. These behave very differently. While a bitmap is a digital representation of your eye, a vector graphic is a visual representation of a mechanical plotter or sign cutting machine.

Vector graphics define pictures as collections of lines, ellipses, triangles, polygons and other basic graphic "primitives." If you use enough of these simple drawing elements, you can create very sophisticated pictures. However, vector graphics are limited to storing mechanical art – they cannot handle photorealistic subjects because there's no software sufficiently sophisticated to be able to look at a photograph and draw a realistic representation of it using lines and filled areas. This is why graphic artists still get paid big bucks.

Both bitmap and vector graphics have their uses – they're just not interchangeable. It's actually possible to convert vector graphics to bitmaps – except for very simple bitmapped graphics, it's not practical to convert in the other direction.

Vector graphic formats include Encapsulated PostScript files, which use the extension EPS, Corel Draw files, which use the extension CDR, and AutoCAD drawings, which use the extension DXF. Graphic Workshop does not really support these formats, nor is it likely to in the foreseeable future.

To further confuse this issue, some vector graphic formats can include bitmapped elements. For example, you can store bitmapped objects in a Corel Draw file. Graphic Workshop does support some of the vector formats in the very limited sense of being able to extract their bitmaps and convert these bitmaps to more conventional formats.

This is a somewhat esoteric issue, and one you can probably safely ignore.

Matters of Colour

Photorealistic images look best if they're stored in a format which allows them to be reproduced with as many of the 16,777,216 colours a computer can display as they feel like using. As has been discussed, this can leave you with very large files. In addition, it can waste a lot of real estate if you want to store something other than a photorealistic image in such a file.

For practical purposes, the next step down from a file which can store images having a maximum of 16,777,216 colours, is one which can store images having a maximum of 256 colours. These latter files are referred to as using "palette-colour." The colours in a palette-colour file are derived from a potential palette of 16,777,216 colours, but no more than 256 of them can be used in any one image. Because of the way palette-colour files are structured, the amount of space required to store an image in one will be a third of that required to store the same image in a true-colour file.

Palette-colour files have an inherent problem – 256 colours are not sufficient to represent a photorealistic image so it looks photorealistic. There's a way to cheat around this problem – photorealistic images can be dithered to 256 colours to create a convincing simulation of all their colours. Dithering involves using patterns of alternating coloured dots to simulate more colours than are actually available.

This is what a photorealistic image looks like before and after dithering – notice the spots on the mermaid. The photorealistic image, on the left, will only look photorealistic if you have a Windows screen driver capable of displaying more than 256 colours. If it looks ugly, see the Drivers document installed with Graphic Workshop.

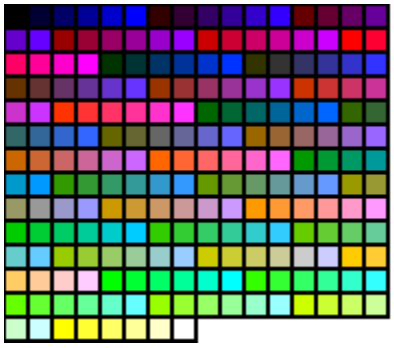
While dithering manages the digital equivalent of squeezing a one-gallon cat into a one- quart mason jar and tightening the lid, it does so at a price. Dithered images don't look as nice as the true-colour originals from which they're derived. They exhibit pronounced aberrations if you resize them. Finally, dithering creates precisely the sort of images that compress very badly.

Palette-colour files are useful for storing line art, that is, graphics which consist primarily of solid lines and areas of colour. The lossless compression typically applied to palette-colour images works well on these sorts of graphics.

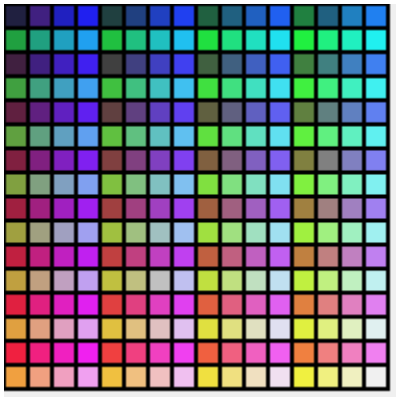
The number of colours used in a palette-colour file can be 2, 4, 8, 16, 32, 64, 128 or 256. A two-colour file is typically black and white, although in practice any two colours can be used. Only 2, 16 and 256 colour palette-colour files are usually found, and several of the common palette-colour file formats only support these colour depths. If you were to convert an image with 32 colours into the BMP format – on of the formats which only supports 2, 16 and 256 colour palette-colour images – it would be promoted to 256 colours.

There are a number of conventions surrounding palette colour. While palette-colour images can use whatever palettes they like, there are several default palettes which turn up, especially on the internet. This is the standard Windows palette, the sixteen colours that every Windows system can always display no matter how its screen driver is configured.

This is the Netscape "safe" palette. If you run a web browser on a system with a 256-colour Windows screen driver, all the graphics displayed on the web pages you surf will be adjusted to use colours from this palette. The results can look pretty grisly.



Finally, this is the 256-colour "orthogonal" palette – this is the palette created when an even dispersal of colours from back to white is generated.



If you have Graphic Workshop dither an image from true-colour to palette-colour, it will by default "quantize" a new palette. This means that it will create a palette of 256 colours which best represent all the colours in the source image. As such, left to its own devices, every palette-colour image it creates in this way will have a unique palette. This can lead to some unexpected side effects if you attempt to combine two 256-colour images and store the results

in a file format which only supports 256 colours. While each image by itself only has 256 colours, the two images together will probably have more than 256 colours.

One of the most common sources of palette problems is the GIF format, which is widely used and only supports a maximum of 256 colours. Attempting to "convert" a JPEG image, which supports 16,777,216 colours, to GIF, is somewhat impossible for this reason. Graphic Workshop provides a way around this problem, of course – the discussion of GIF and JPEG files later in this document will prove useful in this respect.

Understanding Bitmaps

You can save yourself a lot of head bashing if you have a clear understanding of what bitmaps really are before you start working with Graphic Workshop. The first important issue is that of bitmap size and dimensions.

Bitmaps have dimensions, but they don't have sizes.

As was touched on earlier, a bitmap is a matrix of coloured dots. We can define the dimensions of a bitmap as the width and depth of the matrix. However, the dots themselves have no particular size. Because it's impossible to know how big one dot is, the size of a bitmap in inches or centimeters cannot be determined.

When a bitmap is displayed by a specific device, such as your monitor or a printer, each dot appears as an entity with dimensions, and as such the bitmap as it's displayed will have real, measurable dimensions. However, this only specifies the size of a bitmap as it appears on a particular device. A 640 by 480 pixel bitmap will have the dimensions 8.53 by 6.40 inches if it's displayed on a 17-inch monitor – essentially a 75 dot per inch device. The same bitmap will have the dimensions 2.13 by 1.60 inches if it's printed by a 300 dot per inch laser printer. In the latter case, the bitmap didn't get smaller – its individual pixels did.

Higher resolution bitmaps are those which have larger dimensions in pixels. If you scan a graphic into a 75 dot per inch bitmap and 300 dot per inch bitmap, the latter will use more pixels to represent the same details, and as such the details will be more accurately depicted. Note, however, that doubling the resolution of a bitmap will quadruple the number of pixels involved, and hence the file size.

Understanding how bitmaps work will help you to better use the features of Graphic Workshop which serve to modify images. For example, one of the most common modifications applied to bitmaps is modifying their pixel dimensions. Graphic Workshop refers to this as "scaling." Other applications call the same function "resizing" and "resampling."

As we've seen, you cannot change the apparent size of a bitmap by changing the size of its individual pixels, because pixels have no specific size. To make a bitmap appear to occupy more of your screen or a larger area on a printed page, it must be scaled up. In a simple sense, all its pixels must be moved apart and new pixels added to fill in the gaps.

To make a bitmap appear smaller, some of its existing pixels must be discarded and the remaining ones moved closer together.

Scaling always affects the appearance of a bitmap at least a bit. In scaling a bitmap up, the new pixels can be duplicates of the existing pixels or intermediate colour values derived from a transition between the colours of the existing pixels, what Graphic Workshop calls "antialiased" scaling. The latter can produce surprisingly good results, but a scaled-up bitmap will never look as good as a bitmap created at the resolution you needed to begin with.

In scaling a bitmap down, details in the source image will be irretrievably lost in the destination image.

Bitmaps can also be rotated. If you rotate a bitmap by an even multiple of 90 degrees, the rotated image will experience no loss or degradation, and were you to subsequently rotate it back to its original orientation it would be indistinguishable from its source. See the Transform function of Graphic Workshop for these simple rotations.

Rotating a bitmap by increments of less than 90 degrees, what Graphic Workshop calls "free" rotation, requires that some of the source pixels be juggled to find their correct locations in the rotated image. As such, a modest degree of image degradation may be introduced into free rotated images. Free rotation is handled through the Filter function in the Graphic Workshop View mode.

It's important to keep in mind that the sorts of image degradation mentioned herein is much more noticeable in low resolution bitmaps than in higher resolution ones. As a quick rule of thumb, a low resolution bitmap is one which you can more or less view in its entirety on your monitor without any zooming in the graphic Workshop view mode – that is, an image with pixel dimensions of 1280 by 1024 or less. The Get Info function of Graphic Workshop can be used to ascertain the pixel dimensions of the bitmap in any graphic file.

GIF Files

The GIF format is one of the most commonly used graphic file formats, especially on the internet. It's worth knowing a bit about it, as it's not always what it seems to be.

As it appears on the world wide web, the GIF format is exceedingly useful in that it can contain animations. Its internal structure is such that it can store multiple images and the controls to make them appear as real time animation. If you attempt to view such a graphic with Graphic Workshop, it will be displayed as a real time animation. The GIF Construction Set UltraLight application that comes with Graphic Workshop Professional can be used to create such animations.

The principal catch in using GIF files is that the GIF format is only capable of supporting a maximum of eight bits of colour, or 256 colours in all. This means that you cannot convert directly from a 24 bit file, such as a JPEG, to the GIF format. If you attempt to do with Graphic Workshop, you will encounter a message that says "too many colours."

To perform such a conversion with Graphic Workshop, use the colour reduction option of the Effects function. This will dither your source true-colour image down to 256 colours. The 256-colour Quantized palette option will produce the most attractive results. Note that reducing a true-colour image in this way will usually leave you with acceptable images, but they will not be as detailed as the source true-colour graphics they were derived from. This is something you'll have to live with if you want to store pictures as GIF files.

It's also important to keep in mind that while Graphic Workshop, GIF Construction Set and most web browsers can correctly display multiple-image animated GIF files, very few other applications can do so. If they read GIF files at all, most graphic packages only know how to deal with the first image in each file.

The other curious aspect of GIF files to be aware of is their legal status. This will not affect you directly as a user of Graphic Workshop, all rumours to the contrary not withstanding. Having said this, please note that the following is based on our current understanding of the legal status of GIF files, but it does not constitute legal council. Our lawyers made us say that.

The GIF format was originally created by CompuServe. The image compression algorithm used by GIF files is called LZW. At the time GIF was created, back in 1987, CompuServe appears to have assumed that LZW was a public domain entity – at least, there's no indication that they did a patent search to find out whether LZW was owned by anyone. CompuServe announced the GIF format in 1987 with the following grant of rights to developers:

"While this document [the GIF format specification] is copyrighted, the information contained within is made available for use in computer software without royalties, or licensing restrictions."

As it happens, CompuServe didn't actually have these rights to grant – LZW is a patented entity which is currently owned by the Unisys Corporation.

Unisys actually acquired the LZW patent in 1985. They didn't make much noise about it until December of 1994, however, leaving sufficient time for GIF to be widely adopted as a graphic file format. Among other things, it became the de facto standard for lossless graphics on the web.

At the end of 1994, Unisys announced that it would be demanding royalties from any developer who created for-profit software which can read or write graphic files using LZW compression – for practical purposes, this includes GIF and some TIFF files.

Our understanding of this situation, as explained to us by our trademark and patent attorneys, is that Unisys can demand a royalty for software which uses its patented algorithm to read or write GIF files, but not for the data the algorithm creates. As such, while we must pay Unisys a royalty on each copy of Graphic Workshop registered, you're safe in using GIF files without any interference from Unisys.

Our attorneys have also recommended that we not say what we think about Unisys' conduct in this situation, so we'll leave it to your imagination. Note that Unisys would now like to sell us all Internet-based services and hardware. Ya, right...

JPEG Files

The JPEG format, as has been touched on earlier, uses "lossy" compression to get more graphics into a smaller file than would otherwise be possible. The use of lossy compression bespeaks an admirable degree of sneakiness, and can be genuinely useful if you apply it correctly. There are a number of things that may not be apparent in using JPEG files, however, and which might make your use of them less than optimum.

People who use JPEG files without knowing what's really going on usually develop serious personality disorders with time, and frequently find themselves in court for twisting the heads off small dogs for no easily-explained reason.

Note that most of the information in this discussion also applies to ART files and to the Fractal Image FIF, PIC, KDC and FlashPix formats which can be read by Graphic Workshop. These are all lossy-compression graphic file formats.

To begin with, an image written to the JPEG format will be degraded at least a bit. The amount of degradation, the "quality factor," can be set using the JPEG/ART Quality Factor control in the Setup dialog for Graphic Workshop. If this is set to 100, almost no degradation will occur when an image is written to a JPEG file. Of course, the resulting file will not be compressed very much. If it's set to a value close to zero, the resulting image will be all but unrecognizable, although it will fit into a very small file indeed. The default value of 75 is usually a good compromise.

The amount of image degradation caused by writing a graphic to a JPEG file will be a somewhat subjective matter. Especially if you're creating JPEG files for an application in which file size is important, it's often worth experimenting with the quality factor to see how low you can set it for a particular image and still achieve an acceptable level of image quality.

Because the JPEG format always degrades image to some extent, it is not a good choice for archival image storage or for storing high resolution scanned images. Use the PNG format to store graphics you want to preserve the quality of, and convert them to JPEG when you specifically need the extra compression the JPEG format offers.

Secondly, the image degradation caused by the JPEG format is cumulative. As such, if you write an image to a JPEG file, and then read it from the JPEG file and write it back to the JPEG format, it will have suffered two passes of image degradation. For this reason, it's a supremely bad idea to open a JPEG file in an image editor, make changes to the image and save it back to JPEG. The same holds true for the Filters function in the View mode of Graphic Workshop.

You should store your images in a non-lossy format if you plan to modify them, and only write them to JPEG when you're done changing them.

The JPEG format can only store 24-bit colour images. If you store an eight-bit image into a JPEG file, such as a picture from a GIF file, it will be converted to a 24-bit image before it's

stored. This will also be true of images with even less colour depth. Doing so rarely improves on the files sizes in question.

The compression used by the JPEG format works well on photorealistic images, wherein the type of detail reduction it performs is often unnoticeable. It works very, very badly on text, line art or other types of mechanical graphics, which it will degrade quite noticeably. These sorts of graphics should be stored in another format, such as GIF, BMP or PNG.

Unlike GIF, JPEG does not support transparency or multiple images. It cannot be used for animation.

There are two types of JPEG files extant as of this writing, called "sequential" and "progressive". A sequential JPEG file stores its image as a simple bitmap. A progressive JPEG files stores its image such that it can appear initially out of focus when it begins to download to a web page, and resolve itself as more of the image is received by your web browser. This makes little difference to Graphic Workshop, which can read both types.

Very few applications other than Graphic Workshop and web browsers can open a progressive JPEG file. There's a switch in the Graphic Workshop Setup dialog called Write Progressive JPEG. Do not enable this option unless you have a particular need for progressive JPEG files.

While colour JPEG files always support true-colour, for historical reasons the JPEG library knows how to make them look like 256-colour images. In effect, it can dither them as they're unpacked. This option is useful if you want to display a JPEG file on a system with a 256-colour display driver – not a likely occurrence under Graphic Workshop. The resulting dithered images typically look grainy and unattractive as compared to their true-colour parents. There's a switch in the Setup dialog of Graphic Workshop called Read JPEG/FIF as RGB. It should be enabled unless you can come up with a reason to have Graphic Workshop treat JPEG files as 256-colour images.

While they're fairly rare, a few "grey scale" JPEG files exist on the net. These are images rendered in 256 levels of grey, rather than in 16,777,216 colours. Graphic Workshop will read these correctly, but if you write one back to a JPEG file, it will be stored as a colour image in which all the colours are grey.

TIFF Files

The TIFF format is a by now somewhat venerable graphic file format – with a few nasty tricks. It was designed to always be state of the art by having lots of room for expansion. Whenever a new sort of image compression appeared, such as JPEG, it could be incorporated into the TIFF format too. TIFF files can be stored to use the internal number structure of any type of microprocessor. They can be made to store all sorts of information about the pictures they hold.

The problem with all this flexibility is that while the TIFF format itself can learn new tricks, existing software which reads or writes TIFF files doesn't always keep up with them. In fact, because a TIFF reader that knows how to correctly interpret the by now thousands of variations on the TIFF format is a huge undertaking, most applications which purport to be able to open

TIFF files can really only open a defined subset of them. It's possible for one application to write perfectly correct TIFF files and a second application not be able to open them, even if it embodies valid TIFF support. In fact, its very common.

Graphic Workshop can read a very large number of TIFF variations, and we periodically upgrade its TIFF reader as new mutants appear. It's not beyond imagination that you might encounter a file which does not read, however. Do let us know if you find one.

One common problem with TIFF files is two-colour TIFF documents which read correctly, but are reversed, like a photographic negative. This is caused by an error in the way these TIFF files are structured. The Reverse function of Graphic Workshop can be used to store them correctly.

Graphic Workshop can write a number of types of TIFF files – the TIFF Write Options combo box in its Setup dialog will configure it to create TIFF files with each of the available compression types supported by TIFF as of this writing. The Reference document installed with Graphic Workshop includes a complete discussion of these options. Because the authors of some software which opens TIFF files are less than wholly forthcoming about the types of TIFF files their applications can handle, you might have to experiment with these options to create a suitable TIFF file for the software you'll be using.

The best thing you can do with TIFF files is to avoid them as much as possible.